Towards a Stable Decision-Making Middleware for Very-Large-Scale Self-Adaptive Systems.

Russel Nzekwa, Romain Rouvoy, Lionel Seinturier

Towards a Stable Decision-Making Middleware for Very-Large-Scale Self-Adaptive Systems

Russel Nzekwa[a], Romain Rouvoy[a], Lionel Seinturier[a]

[a]*ADAM project-team*
University of Lille 1, LIFL - UMR CNRS 8022, INRIA Lille - Nord Europe
59650 Villeneuve d'Ascq France

Abstract

With the development of the communication infrastructures, the number of applications collaborating at large scale increases. To maintain, and continue to deliver services of good quality to the end-users, very-large-scale applications continuously adapt themselves, depending on the changes in their surrounding. Stabilization of the system thus becomes a keystone issue in the adaptation process, in order to reduce the system reconfiguration cost. Existing approaches, for the stabilization of very-large-scale systems, provide solutions that are partially efficient. For example, learning-based stabilization algorithms give good results in predicting application behaviors, but still suffer from their weak reactivity. In this paper, we propose an approach of combining different goals-oriented stabilization algorithms, in order to provide sustainable and efficient stabilization for large-scale systems.

Keywords: context-awareness, self-adaptation, stabilization, very-large-scale systems

1. Introduction

Because of the huge amount of data that they are intended to process, very-large-scale systems need to retrieve only relevant information from the data flow, in order to perform the system adaptation. To optimize the system adaption process, stabilization mechanisms are required. The concept of context region defined in *Quality of Object* (QuO) [1], is an example of a stabilization mechanism that advocates a strict partition of context space, to reduce the adaptation side effects. However, context regions do not handle application behavior during the transition of the system from one state to another. On the contrary, some stabilization algorithms based on learning strategies, such as *Bayesian Network* (BN) are much more efficient in handling that issue. But, despite some good results in predicting application behavior, even during the state transition, they still suffer from weak reactivity. The stabilization process can be more efficient, if the stabilization mechanisms can be adapted to the processed data. We call *stabilization mechanisms*, the set of algorithms or techniques aiming at regulating the responsiveness of adaptive system. This paper describes an adaptive stabilization approach for composing stabilization algorithms in order to increase their efficiency and accuracy.

The rest of the paper is organized as follows. Section 2 presents the motivations and challenges of the stabilization for very-large-scale systems. Section 3 discusses some related works. Section 4 describes our stabilization approach. We show some simulation results in Section 5. Finally, we conclude in Section 6.

2. Motivations & Challenges

Motivations With the growing complexity of software nowadays, the cost (time, money) of maintaining applications in a changing environment increases exponentially. Therefore, there is a high demand for self-managing system

Email addresses: russel.nzekwa@inria.fr (Russel Nzekwa), romain.rouvoy@inria.fr (Romain Rouvoy), lionel.seinturier@inria.fr (Lionel Seinturier)

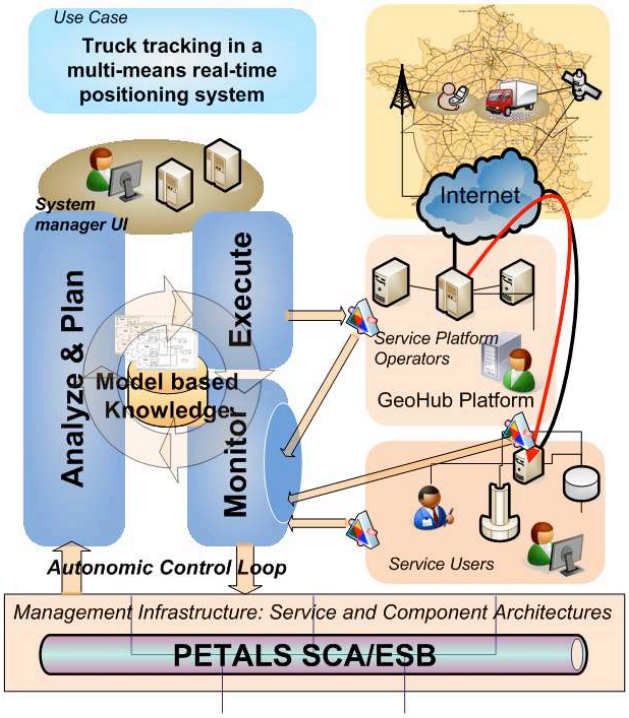

Figure 1: Self-Adaptive Distributed Application

achieving desired quality requirements with a reasonable cost. Self-adaptive software answers to this demand. Such systems are able to *monitor*, to *detect significant* changes, to *decide* how to deal with, and to *act* consequently. When changes in the application environment frequently occur, it is important to limit system reconfigurations in order to keep a good tradeoff between the adaptation cost and the system performance. This is specially true, when dealing with large-scale self-adaptive applications where the communication cost can be expansive. On this specific issue, stabilization mechanisms play a key role allowing to reduce adaptation cost. As an example of a motivation scenario, and to easily point out challenges of the stabilization issue for large-scale self-adaptive application, we consider the "Trucks Tracking" scenario. This scenario is the subject of SALTY ANR project [1].

Trucks Tracking scenario: *We consider a self-adaptive application for the management of the truck fleet (800.000 to 1.000.000 trucks), of a company specialized in the transport of fragile products. All the trucks do not have the same characteristics. Some are equipped with good air conditioning system, while other provide robust slip system. In the same way, transported goods do not have the same requirements. Some are very sensitive to the temperature variation, when other need high security system. The overall objective of the system is to make sure the trucks reach their destination on time. The application must be able to detect stop times, temperature variation inside containers, security violation access and truck position. The application is also connected to remote services like the weather service or the city traffic service. Self-adaptive application, must also notify destination logistic platforms about truck arrival. If one truck gets into trouble, the system can remap the route of other trucks to help it. All aforementioned information is sent to the central location platform which processes and decides which adaptation process can be triggered.* Figure 1, provides an illustration of the scenario.

From the analysis of data exchanged between the nodes of the application, several adaptations can result, such as, the frequency of the positioning requests, or the level of allocated resources. Given the large amount of data involved, and the distributed nature of the application, a naive technique consisting of sending all data for analysis cannot be effective. Therefore, stabilization mechanisms must be implemented.

Challenges In our opinion, two important challenges are to be met, when dealing with stabilization of self-adaptive applications in a large-scale distributed environment. The first challenge is to find the right level of stabilization, so

[1] SALTY ANR project: a research project funded by the french research agency. `http://salty.unice.fr`

that the application can be reactive enough to reflect important variations in the surroundings. Secondly, when self-adaptive applications are running, new features (data, services) can be added to fill specific needs. In this particular case, to be efficient, the stabilization mechanism need to be adapted in order to take into account the new added features. We believe that an adaptive strategy for the composition of stabilization algorithms is the beginning of the solution to move towards more accurate and efficient stabilization of self-adaptive system. In the next section, we present our proposition on this issue.

3. Related Work

Stabilizations techniques are used in the project TEA [2]. The principal limitation of this project is that implemented stabilization techniques are tightly coupled to application architecture and cannot easily be replaced or modified. [3], presents the PHOENIX project which targets the management and monitoring of distributed system (clusters) processing many heterogeneous data. To improve the performance of the system monitoring, and detect emergency situation just when they happen, PHOENIX introduces delta operator (DO) with other first order logic operators. DO allow evaluation of signal amplitude variation, so the data flow is relayed only when it reach the required threshold value. Despite good results recorded by the method, one of the main drawbacks is that the PHOENIX platform does not support extension of new operators, and that the stabilization method is targeted for quantitative numerical data. In [4], Padovitz suggested an architecture to handle stabilization issue in self-adaptive systems on the base of Kalman's filter learning algorithm. The main limitation of that approach was the weak reactivity of the system compared to other approaches.

From the "Truck Tracking" scenario presented in the previous Section 2, we can notice the following points concerning the system reconfiguration.The amount of information to process for the geographical localization of trucks, is huge and variable: (i)- the number of trucks to track , (ii)- variable frequency of the requests , (iii)- variable precision of the positioning information. In order to provide a good service, the system must implement dynamic adaptation mechanisms. Implementing learning-based stabilization algorithm like Dempster-Shaffer (DST) [5] or Bayesian Network (BN), will improve the knowledge of the system about the environment, but at the same time will introduce latency in detecting new changes. The method proposed by Padovitz et al. [4], in the case of the scenario is not suitable, because of the complexity of the mathematical expression determining the instability of the system, which increases with the number of arguments.

To the best of our knowledge, a unique stabilization algorithm that could meet all requirements of self-adaptive applications does not exist. In fact, in the context of very-large-scale systems, the stabilization strategies require to be flexible enough in order to adapt to the evolution of the environment. We believe that, in the perspective of improving the management of stabilization process for self-adaptive systems, a solution can be found in the flexible combination of several existing approaches.

4. Flexible Context-Aware Architecture

This section presents our stabilization approach for self-adaptive systems. We consider that in order to boost flexibility and efficiency, stabilization issue should be handled separately from decision-making basics concerns. For the general case of an autonomic control loop [6], we suggest to integrate stabilization blocks at two levels of the chain processing information data: Firstly, after acquisition of raw data from different probes or sensors, and secondly, after the decision-making block. Figure 2 depicts this architecture. The purpose of the first stabilization block, located between data collectors and decision-making blocks, is to filter data coming from collectors and to forward to decision-making only significants values that could lead to system reconfigurations. This considerably reduces access to decision-making module, resulting in a gain of performance for the system, since running decision-making processes is quite often a costly procedure for the application. The purpose of the second stabilization block, located in our architecture between decision- making block and actuator, is to optimize the work of actuators by finding the suitable moment when reconfigurations of the system can be executed. This issue has a significant impact when dealing with distributed applications, where the decision-making structure and the application are not hosted on the same machine.

In order to meet flexibility requirements for an efficient stabilization of self-adaptive applications, we propose to define a model dedicated to the composition of stabilization algorithms.

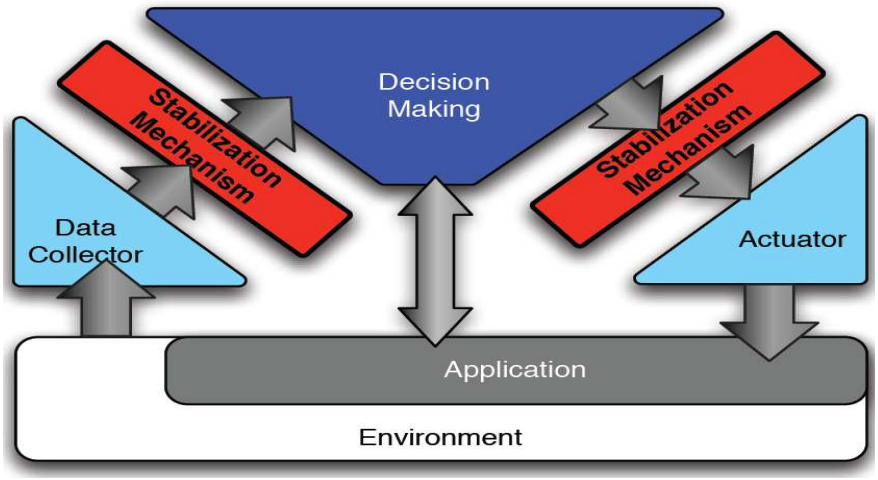

Figure 2: Flexible context-aware architecture

4.1. Composition Model

Our composition model defines how to combine different stabilization mechanisms in order to meet flexibility in the application. The model consists of two modalities of composition: *Horizontal* and *Vertical composition*.

4.1.1. Horizontal Composition

Learning-based stabilization algorithms, like Dempster-Shaffer (DST) [5] or Bayesian Network (BN), while providing a good efficiency in predicting contextual changes, introduce a latency in detecting variations in the application environment. This weakness can be filled in, by associating to learning-based algorithms other algorithms with a higher reactivity speed.

Horizontal composition consists in executing concurrently several stabilization algorithms. The idea behind this concept is to benefit from passive and reactive stabilization techniques by combining them adequately. The detection of irregular application's behavior can be done by combining a reactive algorithm like Delta Operator (DO) , and less reactive algorithm like Dempster-Shaffer (DST) algorithm, through a composition rule (CR). A CR can be some simple rule like " $max(v_n, v_{n+1}) \Rightarrow proceed \quad value$", where v_n, v_{n+1} are context values, or a more complex rule involving Quality of Context (QoC) of the processed data. Hence, using this composition model can help to improve accuracy of the stabilization process while keeping a reasonable level for the reactivity of the system. We will further detail to this example in section 5.

4.1.2. Vertical Composition

Vertical composition in our approach consists in applying sequentially two or more stabilization algorithms on the same data set. Some works [7], suggest that it can be interesting in terms of performance for stabilization of context information to apply successively several algorithms of stabilization on the same sample of data. A good illustration of that idea is the work of Sekkas et al. [7] where the authors compare the efficiency of using Bayesian Network (BN), Dynamic Bayesian Network (DBN), and Fuzzy Logic (FL) alone or in a combined way. In our approach we believe that, the combination of algorithms using *vertical composition* can increase efficiency of the stabilization. In order to limit the overhead introduced by the use of several algorithms, "cheap" (low execution cost) algorithms are found at the beginning (bottom) of the stabilization chain while "expensive" (high execution cost) algorithms are on the top of the architecture. This association rule is mostly justified by the fact that, the amount of processed data decreases from the bottom to the top, thus more costly algorithms at the top of the architecture would have to process less data, decreasing by the same way the overall cost of the stabilization process, which is tightly bound to the amount of context information processed.

Figure 3 gives an illustration of the composition model. From left to right, we have horizontal composition, then vertical and finally the combination of the both types of composition. Combination of both primitive composi-

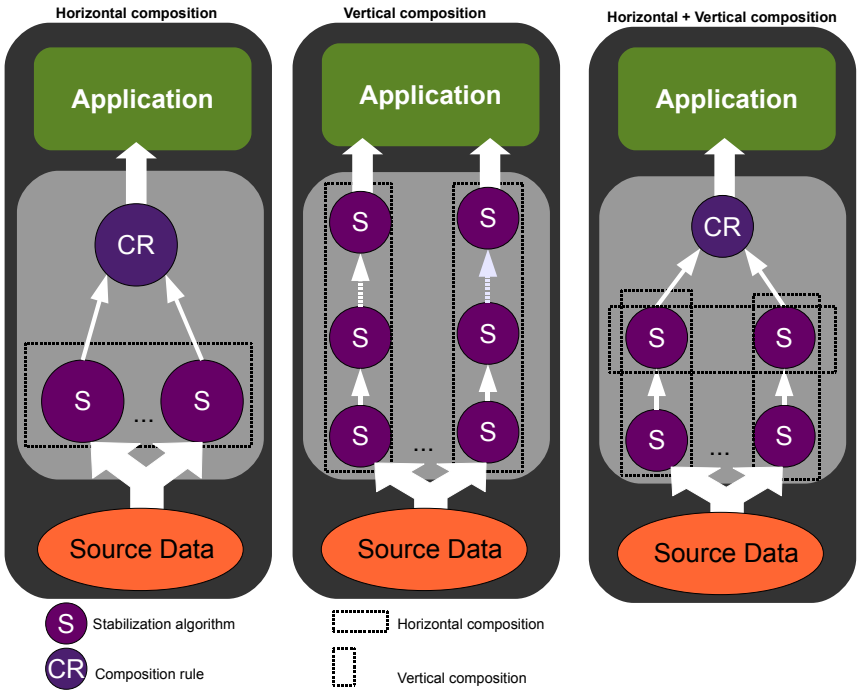

Figure 3: Composition Models

tion models in our approach can be beneficial for the stabilization process in order to meet accuracy and efficiency properties.

5. Evaluation and Simulations

In the previous sections, we presented a novel approach in order to meet flexibility when dealing with stabilizations issues in pervasive applications. We suggested to use composition of algorithms based on the presented composition model. Algorithms composition offers the possibility to maximize the efficiency of the stabilization process at a relative low cost for the application. While in most of the existing systems, the decision-making module applies a generic algorithm to processed data, our approach offer the possibility to adapt the stabilization mechanism to the data being processed. To evaluate our proposition, we have made some experimental simulations that illustrate the feasibility of our approach. Our simulations are based on COSMOS [8](Context Entities Composition and Sharing). We also use the simulation engine SIAFU [9] to generate contextual informations.

The scenario for this simulation is similar to the one presented in Section 2, because the application performs some adaptations depending on the changes in the environment. In the current scenario, we have temperature sensors simulated by the simulator engine that send data concerning the temperature of the environment. Time elapsed between two consecutive readings from sensors is 0.3 second. Collected values have an accuracy close to 0.1. The overall objective of the application process will be to detect all the relevant variations in the application environment, around the range of 24°C to 25.5°C. We consider that a relevant variation in the environment, is a variation that persist along three consecutive readings from the sensors.

The first implemented algorithm, is "Delta operator" (DO) [3]. The choice of the threshold value is motivated by the data sample's variance, in order to choose the most appropriate value. The second implemented algorithm is Kalman filter. The central point for Kalman's filter algorithm is the determination of the matrix of transition A, in this case we defined the transition matrix as following: Given A the transition matrix, given x_{k-1}, z_{k-1}, R the prediction, the value of measured variable at $k-1$ step and the variance respectively. We have defined ε, Δ then A as follows:

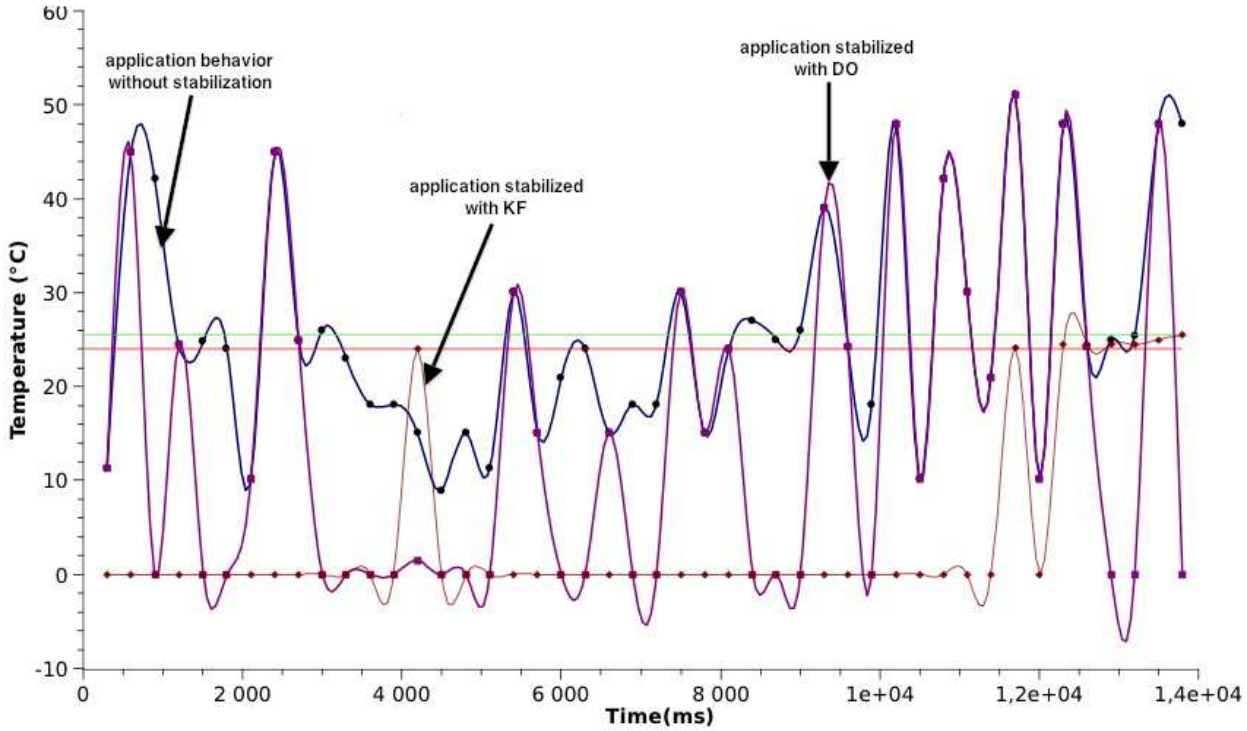

Figure 4: Stabilized application behavior - Kalman filter and Delta operator

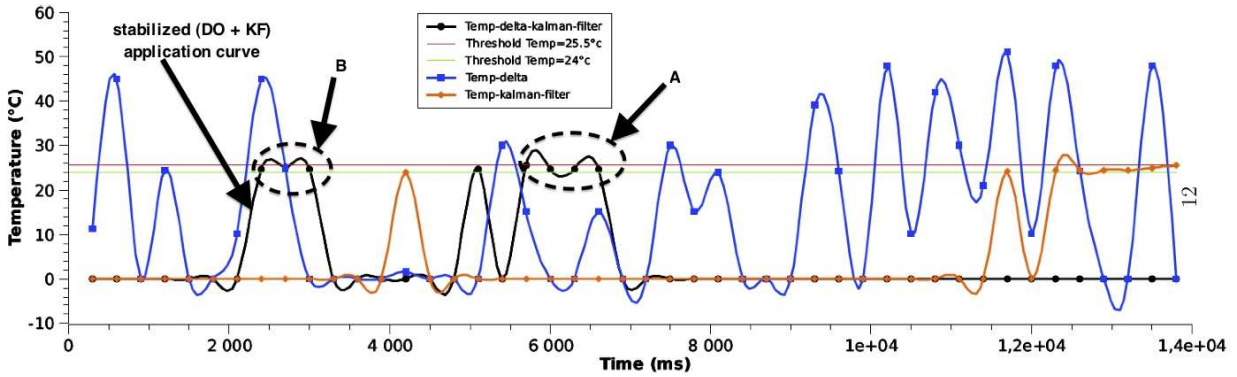

Figure 5: Comparatives curves of stabilized application behavior

$\Delta = x_{k-1} - z_{k-1}, \varepsilon = R * z_{k-1}$

$$A_k = \begin{cases} A_{k-1} & for \quad |\Delta| \leq \varepsilon \\ \frac{z_k - \varepsilon}{x_{k-1}} & for \quad |\Delta| > \varepsilon \quad and \quad \Delta < 0 \\ \frac{z_k + \varepsilon}{x_{k-1}} & for \quad |\Delta| > \varepsilon \quad and \quad \Delta > 0 \end{cases}$$

Figure 4 shows the curves of the application behavior without stabilization, stabilized with KF, and stabilized with DO. As expected, application stabilized with DO is very reactive, we can notice that around $t = 2s$ for example, the curves of the application without stabilization and stabilized with DO are almost indistinguishable. Figure 5 shows application's behavior curves when combining the both algorithms using our model, in this case an horizontal composition. On Figure 5, the curve of the behavior of the application stabilized with both KF and DO algorithms is

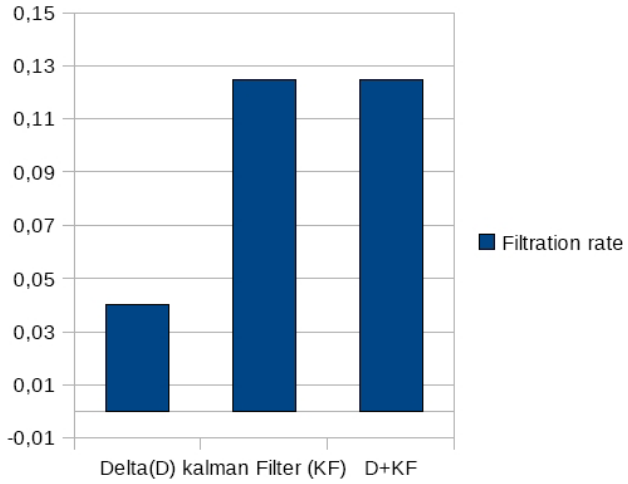

Figure 6: Filtration rate measurements

represented. On that graph, we can notice two things. Firstly, the detection of the first event for the application occurs earlier (zone B) than for an application stabilized by KF only. That is due to the fact that the application benefit from the reactivity property of DO algorithm. Secondly, we can notice that when an event is detected the behavior of the system does not change suddenly (zone A or zone B) even when the next raw value processed by the system does not belong to the target area. That last point is a property that the system inherited from KF algorithm. Obviously, the composition model offers a good compromise between a reactive and less reactive system. Finally, to characterize our system we have measured the filtration rate, the results of the measurement are depicted in 6. We can notice that resulting application (DO+KF) has the same filtrating rate as the application application stabilized only with KF.

6. Conclusion

In this paper, we have proposed an adaptive approach for context stabilization in a distributed environment. For that purpose, we suggest two modalities of composing stabilization algorithms in order to increase accuracy and efficiency of the stabilization. We showed a simulation example to illustrate the feasibility of our approach. In the future, we plan to probe this approach on a real very-large-scale platform like an experimental or a production grid.

References

[1] M. Atighetchi, P. Pal, C. J. P. Rubel, R. Schantz, J. Loyall, J. Zinky, Building auto-adaptive distributed applications: The quo-apod experience, Distributed Computing Systems Workshops (2003) 104–109.
[2] S. Albrecht, W. H. Gellersen, M. Beigl, Multi-sensor context-awareness in mobile devices and smart artifacts, Mobile networks and applications (2002) 341–351Netherlands.
[3] B. Folliot, C. Boutros, X. Bonnaire, Phoenix: a self-adaptable monitoring platform for cluster management, cluster computing 5 (2002) 75–85.
[4] A. Padovitz, A. Zaslavsky, S. W. Loke, B. Burg, Maintaining continous dependability in sensor-based context-aware pervasive computing systems, Hawaii International Conference on System Sciences 9 (2005) 290a, los Alamitos, CA, USA.
[5] H. Wu, Sensor data fusion for context-aware computing using dempster-shafer theory, Ph.D. thesis, Carnergie Mellon University, pittsburgh (2003).
[6] J. O. Kephart, D. M. Chess, The vision of autonomic computing, Computer 36 (1) (2003) 41–50, los Alamitos, CA, USA.
[7] O. Sekkas, S. Hadjiefthymiades, C. Anagnostopoulos, Context fusion through imprecise reasoning, IEEE international conference on pervasive services (2007) 88–91.
[8] R. Rouvoy, D. Conan, L. Seinturier, Software architecture patterns for a context-processing middleware framework, In IEEE Distributed Systems Online (DSO) 09.
[9] M. Miquel, P. Nurmi, A generic large scale simulator for ubiquitous computing, In Third Annual International Conference on Mobile and Ubiquitous Systems (2006) 1–3USA.